Лабораторна робота №3

Криптоаналіз афінної біграмної підстановки

Виконала Годлевська Аліса ФБ-11

Мета роботи

Набуття навичок частотного аналізу на прикладі розкриття моноалфавітної підстановки; опанування прийомами роботи в модулярній арифметиці

Порядок виконання

1. Реалізувати підпрограми із необхідними математичними операціями: обчисленням оберненого елементу за модулем із використанням розширеного алгоритму Евкліда, розв'язуванням лінійних порівнянь. При розв'язуванні порівнянь потрібно коректно обробляти випадок із декількома розв'язками, повертаючи їх усі

2. За допомогою програми обчислення частот біграм, яка написана в ході виконання комп'ютерного практикуму №1, знайти 5 найчастіших біграм запропонованого шифртексту (за варіантом)

3. Перебрати можливі варіанти співставлення частих біграм мови та частих біграм шифртексту (розглядаючи пари біграм із п'яти найчастіших). Для кожного співставлення знайти можливі кандидати на ключ (a,b) шляхом розв'язання системи

4. Для кожного кандидата на ключ дешифрувати шифртекст. Якщо шифртекст не є змістовним текстом російською мовою, відкинути цього кандидата.

5. Повторювати дії 3-4 доти, доки дешифрований текст не буде змістовним.

Хід роботи

1. Реалізую усі необхідні функції для математичних операцій:

```
def greatest_common_divisor(a, b):
    if b == 0:
        return abs(a)
    else:
        return greatest_common_divisor(b, a % b)
```

Знаходимо НСД

```
def extended_euclidean_algorithm(a, n):
    res = [0, 1]
    while n != 0 and a != 0:
        if n < a:
            res.append(a // n)
            a = a % n
        elif n > a:
            res.append(n // a)
            n = n % a
    for i in range(2, len(res) - 1):
        res[i] = res[i - 2] + (-res[i] * res[i - 1])
    return res[-2]
```

Розширений алгоритм Евкліда

```
def modular_equation_solver(a, b, n):
    a = a % n
    b = b % n
    d = greatest_common_divisor(a, n)
    result = []
    if d == 1:
        x = (extended_euclidean_algorithm(a, n) * b) % n
        result.append(x)
        return result
    else:
        if b % d == 0:
            a = a // d
            b = b // d
            n = n // d
            x = (modular_equation_solver(a, b, n)[0])
            result.append(x)
            for i in range(1, d):
                result.append(result[-1] + n)
            return result
        else:
            return result
```

Розв’язання лінійних порівнянь

2.

```
with open("11.txt", "r", encoding="utf-8-sig") as file:
    raw_text = file.read()
text = ''.join(i for i in raw_text if i in alphabet)

bi_list = []
for i in range(0, len(text) - 1, 2):
    bi_list.append(text[i:i + 2])
bi_counter = Counter(bi_list)
sorted_bi_counter = {key: val for key, val in sorted(bi_counter.items(),
                                                     key=lambda ele: ele[1], reverse=True)}
temp_top_bigrams = []
for i in sorted_bi_counter:
    temp_top_bigrams.append(i)
top_bigrams = []
for i in range(0, 5):
    top_bigrams.append(temp_top_bigrams[i])
```

Найчастіші біграми

3.

```
def generate_combinations(frequent_bigrams, cy_bigrams):
    bi = []
    comb = []
    for i in frequent_bigrams:
        for j in cy_bigrams:
            bi.append((i, j))
    for i in bi:
        for j in bi:
            if i == j or (j, i) in comb:
                continue
            elif i[0] == j[0] or i[1] == j[1]:
                continue
            comb.append((i, j))
    return comb
```

Співставляємо

```python
def find_ab_values(combinations):
    ab_values = []
    x1 = index_of_bigram(combinations[0][0])
    x2 = index_of_bigram(combinations[1][0])
    y1 = index_of_bigram(combinations[0][1])
    y2 = index_of_bigram(combinations[1][1])
    a_values = modular_equation_solver(x1 - x2, y1 - y2, 31 ** 2)
    for a in a_values:
        if greatest_common_divisor(a, 31) != 1:
            continue
        b = (y1 - a * x1) % 31 ** 2
        ab_values.append((a, b))
    return ab_values


ab_values_list = []
for comb in combinations:
    temp = find_ab_values(comb)
    if len(temp) != 0:
        for value in temp:
            ab_values_list.append(value)
```

Знаходимо кандидатів на ключ

4.

```python
valid_keys = []
for ab_value in ab_values_list:
    check = 0
    temp_text = decrypt_text(text, ab_value)
    for imp in excluded_list:
        if imp in temp_text:
            check = 1
    if check == 0:
        valid_keys.append(ab_value)
```

Використаю відбракування неможливих біграм (excluded_list):

```
frequent_bigrams = ['ст', 'но', 'то', 'на', 'ен']

excluded_list = ["аь", "еь", "иь", "оь", "уь",
                 "ьь", "ыь", "эь", "юь", "яь"]
```

```
C:\Users\alisa\AppData\Local\Microsoft\WindowsApps\python3.10.
[(703, 956), (703, 956), (703, 956)]
хорошосэрбиллнехотясунулденьгивкарманвотчтобиллвыпростопосеете
```

Текст є фрагментом книги «Кульбабове вино» Рея Бредбері

Висновки: мені вдалося проаналізувати шифр афінної підстановки та дешифрувати текст; також я отримала практичні навички із використанням модульної арифметики.